On Using Populations of Sets in Multiobjective Optimization*

Johannes Bader, Dimo Brockhoff, Samuel Welten, and Eckart Zitzler

Computer Engineering and Networks Lab, ETH Zurich, 8092 Zurich, Switzerland,
firstname.lastname@tik.ee.ethz.ch,
http://www.tik.ee.ethz.ch/sop/

Abstract. Most existing evolutionary approaches to multiobjective optimization aim at finding an appropriate set of compromise solutions, ideally a subset of the Pareto-optimal set. That means they are solving a set problem where the search space consists of all possible solution sets. Taking this perspective, multiobjective evolutionary algorithms can be regarded as hill-climbers on solution sets: the population is one element of the set search space and selection as well as variation implement a specific type of set mutation operator. Therefore, one may ask whether a 'real' evolutionary algorithm on solution sets can have advantages over the classical single-population approach. This paper investigates this issue; it presents a multi-population multiobjective optimization framework and demonstrates its usefulness on several test problems and a sensor network application.

1 Motivation

Most multiobjective evolutionary algorithms (MOEAs) proposed in the literature are designed towards approximating the set of Pareto-optimal solutions [7]. In contrast to single-objective optimizers that look for a single optimal solution, these algorithms aim at identifying a *set* of optimal compromise solutions, i.e., they actually operate on a set problem. With such a set problem, the search space consists of all solution sets and often a set quality measure like the hypervolume indicator [20] is used as a corresponding objective function on sets. From this perspective, current MOEAs can be regarded as hill climbers or $(1, 1)$-strategies on solution sets, cf. [21]. The population represents a solution set and as such one element of the set search space. The usual sequence of operations, i.e., mating selection, variation including mutation and recombination, and environmental selection, serves the purpose of generating a new set; therefore, it can be considered as a (complex) set mutation operator. Since the newly generated population usually replaces the old population without a direct comparison and check, one can speak of a $(1, 1)$-strategy in this context.

The above observation leads to the question of whether the use of an evolutionary algorithm on sets may be beneficial in this multiobjective setting. In other words: can maintaining a population of solution sets in combination with appropriate set selection and set variation operators have advantages over using

* This is an author version of the EMO'2009 paper published by Springer (M. Ehrgott et al. (Eds.): EMO 2009, LNCS 5467, pp. 140-154, 2009). The final publication is available at www.springerlink.com.

a single solution set only? If we consider classical MOEAs as $(1, 1)$-strategies on the corresponding set problem, then we are here interested in extending them to (μ, λ)- or $(\mu + \lambda)$-strategies. To our best knowledge, this issue has not been addressed so far, although there is an interesting close link to parallel MOEAs. Some types of parallel MOEAs, in particular those based on island models, make use of multiple populations that evolve simultaneously and from time to time exchange individuals [15, 13, 1, 16, 4, 14, 5, 18]. However, these approaches can in general not be regarded as full evolutionary algorithms on solution sets, as they usually implement only some aspects of set-based fitness, set-based variation, and set-based selection. The considerations presented in the following are independent of the type of implementation, be it sequential or parallel.

This paper investigates the issue of whether a multiobjective optimizer can benefit from utilizing a population of *solution sets* instead of relying on a single population of *solutions*—mainly in terms of the quality of the generated Pareto set approximation, but also with respect to the computing time. As a basis, we consider the optimization scenario where a set with N solutions is sought that maximizes the hypervolume of the dominated objective space. The specific contributions are:

- A general framework for a population-based evolutionary algorithm operating on solution sets; the framework resembles the island model known for parallel evolutionary algorithms.
- The design of a new recombination operator on solution sets which is tailored to the hypervolume indicator, but the principle of which can be generalized to other unary indicator functions.
- A systematic comparison of the classical MOEA scheme and the multi-population scheme on several test problems with up to four objectives.

The following section provides a brief survey of set-based multiobjective optimization, in particular of hypervolume-based multiobjective search, and a background of related work in the area of parallel evolutionary algorithms. Section 3 introduces our general framework of a multi-population MOEA including the new recombination operator. Section 4 presents and discusses the experimental results, and Sec. 5 contains conclusions and future research directions.

2 Background

2.1 Set-Based Multiobjective Optimization

Given an optimization scenario where: X is the decision space; $x \in X$ denotes a solution or decision vector; k objective functions $f = (f_1, \ldots, f_k)$ are to be minimized; $x \preceq y$ denotes weak dominance of y by x and $x \prec y$ denotes strict dominance[1], the goal is usually to find a set A which represents a good approximation of the Pareto-optimal set. Many ways to assess the quality of a Pareto

[1] A solution x is said to weakly dominate a solution y ($x \preceq y$) if it is at least as good as y in all objectives, i.e., $\forall 1 \leq i \leq k : f_i(x) \leq f_i(y)$. If additionally there exists an objective function f_j with $f_j(x) < f_j(y)$ then x is strictly dominating y ($x \prec y$).

set approximation A exist. One is to consider the space that is weakly dominated by the objective vectors $f(A)$ and bounded by a user defined reference set R:

$$H(A,R) := \{h \mid \exists a \in A \, \exists r \in R : f(a) \leq h \leq r\} \tag{1}$$

Let the corresponding *hypervolume indicator* be the hypervolume of this space $I_H(A,R) := \lambda\big(H(A,R)\big)$, where λ denotes the Lebesgue measure. Of all the numerous measures, the hypervolume indicator or $\mathcal{S}$-metric [3] is one of the most popular; mainly because it is the only known indicator that reflects Pareto dominance, i.e., if a solution set dominates another, the hypervolume indicator of the former is greater than the one of the latter. The goal of hypervolume indicator-based MOEAs can be formalized as finding the solution set A^* that maximizes the indicator value, usually imposing a maximum cardinality of $|A^*| \leq \mu$. By the nature of I_H, the set maximizing I_H is a subset of the Pareto set.

The classical view of MOEAs is illustrated in the upper left corner of Fig. 1. Mating selection, mutation, crossover, and environmental selection operate on single solutions and thereby generate a new—hopefully better—set of solutions. Summarized, one can state that classical MOEAs operate on elements of X and deliver an element of $\mathcal{P}(X)$, where $\mathcal{P}(X)$ denotes the power set of X.

Definition 1. *We refer to an optimizer that operates on elements of the decision space U and returns an element of V as a U/V-optimizer.*

Hence, MOEAs are, from a classical EA perspective, $X/\mathcal{P}(X)$ optimizer. On the other hand, multiobjective algorithms using aggregation are considered as X/X-optimizers. However, the individual steps (fitness assignment, mating selection, mutation/crossover, and environmental selection) of the MOEA, that lead to a modified set, can be abstracted as a set mutation, see the upper right corner of Fig. 1—they are in fact $\mathcal{P}(X)/\mathcal{P}(X)$-hillclimbers [21].

In the following, we propose a general $\mathcal{P}(X)/\mathcal{P}(X)$ evolutionary algorithm as depicted in the lower half of Fig. 1. The question arises, how the corresponding operators (set mutation, set crossover, set mating and set environmental selection) can be created and if they are beneficial for search. To this end, we propose set operators based on the hypervolume indicator. There are already many algorithms using this indicator (e.g., [3, 12]), but they are all $X/\mathcal{P}(X)$-optimizers.

To our knowledge, no study has used the set perspective on evolutionary algorithms explicitly, but parallel evolutionary algorithms can be considered as optimizers operating on sets, as demonstrated in the following subsection.

2.2 Parallel Evolutionary Algorithms

The increasing complexity of large scale problems and the availability of large computer clusters and multiprocessor systems were the first incitement to paralle

The *master-slave* approach uses a master processor that performs all operations on one global population except for fitness evaluations which are delegated to different slave processors [17]. Since this parallelization does not change the

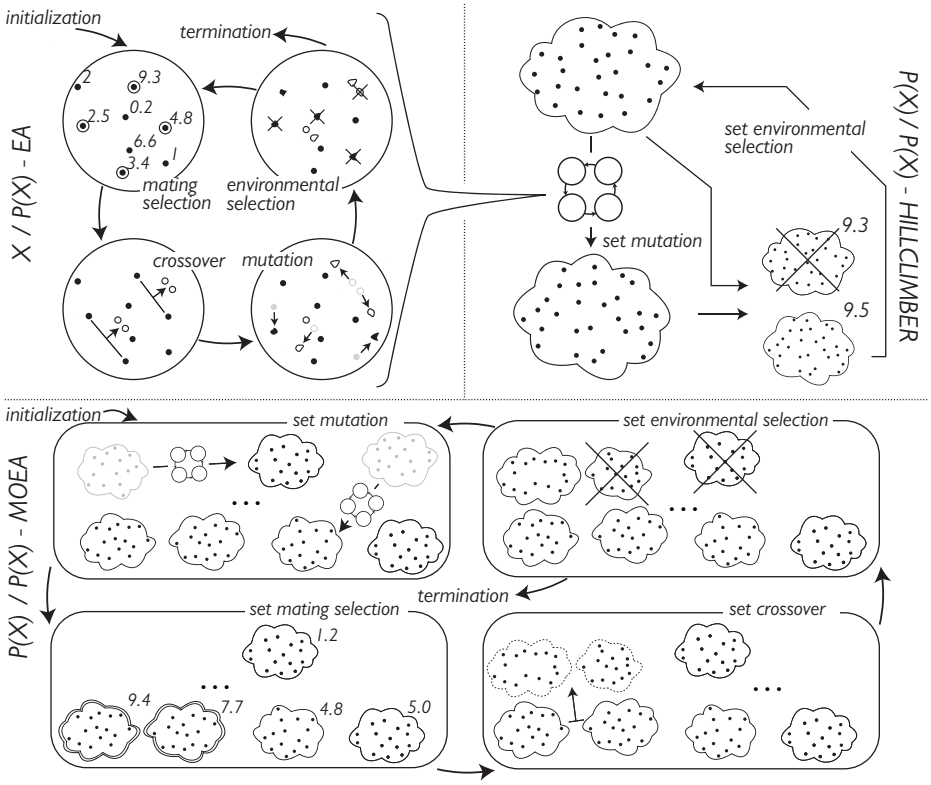

Fig. 1. Illustration of different types of MOEAs: (top left) usual view of a MOEA where the operators work on solutions; (top right) a set-based view of the same algorithm; (bottom) an evolutionary algorithm working on sets, i.e., a $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer.

algorithm itself, master-slave MOEAs can be either seen as $X/\mathcal{P}(X)$-optimizers or interpreted as $\mathcal{P}(X)/\mathcal{P}(X)$-hillclimbers, see Fig. 1.

The second major category of parallel MOEAs—the *island model*—however, can be seen as $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer that use more than one set. An island model MOEA divides the overall population into different islands or independent solution sets. Hence, when abstracting away from parallelization, the island model can be interpreted as an algorithm operating on a population of sets. Each of these sets represents one island which is optimized by a separate EA. This enables running different islands on several computers at the same time.

An island model without any exchange of individuals between islands corresponds to a multi-start approach, where each island represents one run, using different seeds or even different optimization strategies [14]. Such an algorithm mainly benefits from increased robustness of obtained solutions and corresponds to a $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer (see Fig. 1) where each set is mutated and no recombination and environmental selection takes place.

Most island models, however, use a cooperative approach. Although the subpopulations evolve independently most of the time, solutions are exchanged once in a while between islands by *migration*. A well designed migration lets information of good individuals pass among islands and at the same time helps to preserve diversity by isolation of the islands. In contrast to the approaches mentioned above, this paradigm also uses recombination of sets (by migration) and can therefore be advantageous not only in terms of runtime and robustness, but also in terms of quality of the obtained Pareto-optimal solutions [5].

There exist many aspects of migration strategy. (a) The way islands are selected for migration (the set mating selection from a set based perspective) is often done deterministically according to the way islands are arranged [16], where the topology is an important parameter which has to be adapted to the problem structure [15]. (b) The way the population is divided into subpopulations. Often each island corresponds to a different region of the objective space determined manually, e.g., by using cones [4] or by assigning each island to a different subproblem [13]. Instead of explicitly, the division of the objective space into different regions can also happen implicitly, e.g., when distributing the best individuals according to one objective function to different islands [10]. (c) Islands are optimized either by the very same optimizer or by using different parameters. For more details, we refer to [5] and [18].

All island models presented so far do not use the concept of a set-based fitness measure and operators. One exception is the algorithm presented in [1], where islands are randomly selected and both mutation and recombination are applied to subpopulations rather than to single solutions. The quality of the newly generated subpopulations as well as their parents is then assessed by a fitness value and the better sets are kept (set environmental selection). However, the environmental selection only operates locally and the fitness assignment is not a true set fitness since it corresponds to the sum of single fitness values that are determined on basis of a global population.

In this paper, we give first insights on how to use the set-based view on island models to propose a general $\mathcal{P}(X)/\mathcal{P}(X)$ MOEA. In the next section, we systematically investigate which extensions are needed and propose a novel recombination scheme on sets using the hypervolume indicator.

3 A General Framework for a Set-based Evolutionary Algorithm

In this section, we propose a general framework of a $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer for multiobjective optimization the basis of which is a population-based evolutionary algorithm. In contrast to most island-based MOEAs, this new optimizer uses all known operators—mating selection, recombination, mutation, and environmental selection—of a usual evolutionary algorithm, working on sets of solutions. How these operators on sets of solutions can look like is the main focus of this section. In the following, we first describe the general framework and later on present different operators on solution sets.

Algorithm 1 A $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer with $(\mu \dot{+} \lambda)$-selection

Require: number of solution sets in population μ, number of solutions in each solution set N, number of offspring λ, maximum number of generations $g_{\max}$

```
Init: choose population S uniformly at random as μ sets of N solutions from X each
i ← 1 {set generation counter}
while i ≤ g_max do
  M ← ∅
  for all A ∈ S do
    M ← M ∪ {setMutate(A)}
  end for
  M' ← setMatingSelection(M, λ)
  M'' ← ∅
  for all (A_p, A_q) ∈ M' do
    M'' ← M'' ∪ {setRecombine(A_p, A_q)}
  end for
  S ← setEnvironmentalSelection(S, M'')
  i ← i + 1
end while
```

3.1 A $(\mu \dot{+} \lambda)$-EA as a $\mathcal{P}(X)/\mathcal{P}(X)$-Optimizer

Algorithm 1 shows a general $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer that mainly follows the scheme of Fig. 1. The algorithm resembles an island-based MOEA as discussed in Sec. 2.2 with additional mating and environmental selection. Mutation, recombination, and selection on single solutions are considered as mutations on solution sets and the migration operator is regarded as recombination operator on sets.

The algorithm starts by choosing the first population $\mathcal{S}$ of μ sets (of N solutions each) uniformly at random. Then, the optimization loop produces new sets until a certain number $g_{\max}$ of generations are performed. To this end, every set A in the population $\mathcal{S}$ is mutated to a new set by the operator setMutate(A) and λ pairs of sets are selected in the set mating selection step to form the parents of λ recombination operations. Note that the operator "$\cup$" is the union between two multisets; since the population of evolutionary algorithms usually contains duplicate solutions, we also do not restrict the population of Algorithm 1 to sets. In the environmental selection step, the new population is formed by selecting μ sets from the union of the previous population and the varied solution sets. Figure 1 illustrates the steps performed in one generation graphically.

3.2 Mutation of Solution Sets

As mutation operator on solution sets, we propose to use a simple $X/\mathcal{P}(X)$-optimizer with $(N + N)$-selection on single solutions that aims at optimizing the hypervolume indicator of the final solution set directly. This corresponds to a run of a normal hypervolume-based MOEA, as for example [3] or [12], for G generations. The used $X/\mathcal{P}(X)$-optimizer starts with a set of N solutions

that is obtained from the overall $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer's population. For G generations, N solutions of the current set are selected in a mating selection step, these solutions undergo SBX crossover and polynomial mutation as described in [8] and in the environmental selection step, the best solutions from the previous population and the new solutions are selected to form the new population.

The fitness of a solution in the selection steps is a generalization of the hypervolume loss as proposed in [2]. Instead of using the hypervolume that is solely dominated by a single solution as fitness value as it is done in many hypervolume-based MOEAs, e.g., [3, 12], the fitness $I_h^l(a, A, R)$ of a solution $a \in A$ is computed as the expected hypervolume loss if the solution itself and $l-1$ randomly selected other solutions in the population are removed.

Definition 2. *Let A be a solution set, $R \subset Z$ the reference set of the hypervolume indicator, and $l \in \{0, 1, \ldots, |A|\}$ the number of solutions that are to be removed from A. Let $H_i(a, A, R)$, in addition, be the portion of the objective space that is dominated by $a \in A$ and exactly $i-1$ other solutions in A and that itself dominates the reference set R. Then the function*

$$I_h^l(a, A, R) := \sum_{i=1}^{l} \frac{\alpha_i}{i} \lambda(H_i(a, A, R)) \quad \textit{where} \quad \alpha_i := \prod_{j=1}^{i-1} \frac{l-j}{|A|-j} \tag{2}$$

gives the fitness of a solution $a \in A$.

In the mating selection step, we choose $I_h^{|A|}(a, A, R)$ as the fitness in a binary tournament selection whereas in the environmental selection step, we choose $l = N$ since N solutions in the $(N+N)$-selection have to be removed to build the new population. For environmental selection, non-dominated sorting on the set of $N+N$ solutions is performed. Then, non-dominated fronts are added completely to the new population according to their rank until the population size is reached. If the number of selected solutions exceeds N, the solutions with worst fitness are iteratively removed until the population size N is reached again. For a motivation of this new fitness assignment scheme and an evaluation of its usefulness, we refer to [2].

3.3 Recombination of Solution Sets

Since we aim at maximizing the hypervolume indicator in multiobjective search, a recombination operator on sets should also aim at producing offspring with large hypervolume. Therefore, we propose a new recombination operator on solution sets A and B that is targeted at maximizing the hypervolume of the offspring C, see Fig. 2 for an illustrative example. The idea behind the operator is to iteratively delete the worst solution in the first parent and add the best individual from the second parent until no hypervolume improvement is possible. In more detail, the process runs as follows.

In a first step, all solutions in the first set $A = \{a_1, \ldots, a_{|A|}\}$ are ranked according to their fitness $I_h^l(a, A, R)$ in Eq. 2 with $l = 1$ (upper left figure in

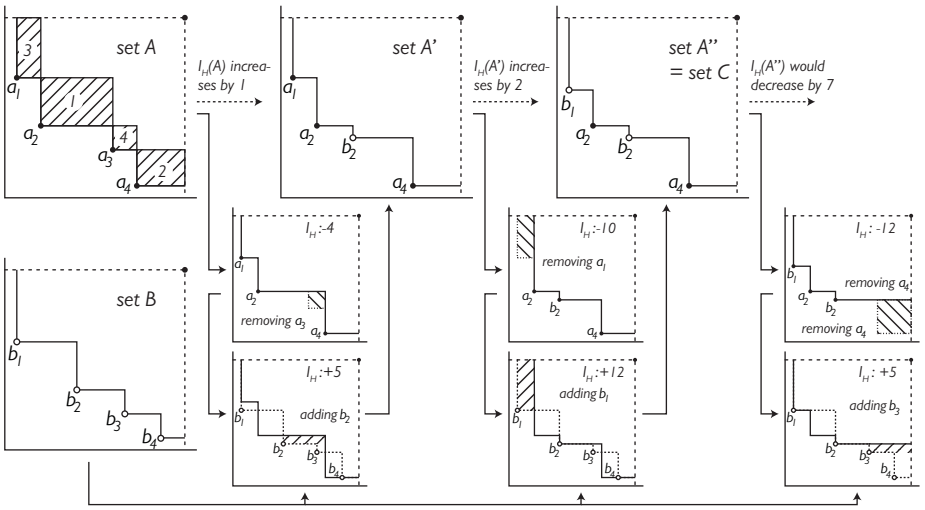

Fig. 2. Illustration of the hypervolume-based recombination operator on solution sets: two exemplary sets A and B with four solutions each are recombined to a set C. First, the solutions in A are ranked according to their hypervolume losses. Then, iteratively, the solution in A with smallest loss is deleted (middle row) and the solution in B that maximizes the hypervolume indicator is added to A (last row) until no hypervolume improvement is possible. For each step, the changes in hypervolume are annotated in the top right corner of the corresponding figure.

Fig. 2). In this case, the fitness of a solution corresponds to the hypervolume that is solely dominated by this solution, in other words, its hypervolume loss. Then, the new set C results from A by iteratively removing the solution a_i with smallest fitness that is not yet removed (ties are resolved randomly, see middle row in Fig. 2) and adding the solution $b \in B$ that maximizes the hypervolume indicator of the new set (last row in Fig. 2). The replacement of solutions stops before the next exchange would decrease the hypervolume of the new set.

Since the fitness values of the solutions in A are only calculated once in the beginning, at most $|A| \cdot |B| + 1$ hypervolume indicator values of $|A|$ points have to be computed in each recombination. Note that the recombination operator can also be seen as a hypervolume-based migration strategy for island model based MOEAs where each island obtains solutions from a neighboring island as long as its hypervolume increases. Another important aspect, we would like to mention is the asymmetry of the recombination operator, i.e., $\text{setRecombine}(A_p, A_q) \neq \text{setRecombine}(A_q, A_p)$. This asymmetry is the reason for selecting ordered pairs in the set mating selection step of Alg. 1.

3.4 Mating and Environmental Selection

In the following, we present four different variants of mating and environmental selection combinations. Two variants choose sets for recombination directly from

the mutated sets (we call them A-variants) whereas the other two variants choose one mutated set as the first parent and the set containing all solutions of all other sets as the second parent for recombination (called B-variants):

Variant A1 randomly selects μ pairs of sets in the mating selection step and uses (μ, μ)-selection in its environmental selection step.

Variant A2 selects all possible $\mu \cdot (\mu - 1)$ pairs of sets in mating selection and selects the best μ out of the $\mu \cdot (\mu - 1)$ new sets in the environmental selection.

Variant B1 selects one pair of sets only, where the first set $A_1 \in M$ is selected uniformly at random and the second set A_2 is chosen as union of all $A \in M$ except A_1 itself. In the environmental selection step, variant B1 copies the only new set μ times to create the new population of μ identical sets.

Variant B2 selects μ pairs of sets by choosing every set of M once as the first set A_1 of a parent pair and the second set A_2 of the pair is chosen as union of all $a \in M$ except A_1 itself as in variant B1. The environmental selection of variant B2 chooses all μ newly generated sets to create the new population.

Note that all variants perform the mating selection independent of the hypervolume indicator the consideration of which may improve the optimizer further. Note also that parallel MOEAs, when interpreted as $\mathcal{P}(X)/\mathcal{P}(X)$-optimizers, usually do not perform environmental selection and select the individuals for mating according to a fixed scheme given by the neighborhood of the islands.

4 Experiments

The experiments described in this section serve three main goals. First, we extensively compare four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants with a standard MOEA on various test problems. Then, we study the set mutation operator, in particular, the length G of a set mutation step. Third, we apply the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer to a sensor network application and compare it with the standard MOEA.

4.1 Experimental Setup

As the baseline standard MOEA, we use the algorithm described in Sec. 3.2 and [2]. Single solutions are mutated by polynomial mutation and recombined via SBX crossover [7]. In addition, we consider four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants A1, A2, B1, and B2 named after the used selection scheme as described in Sec. 3.4. The set mutation and set recombination operators are the same in all variants and implemented as described in Sec. 3. For a fair comparison, the standard MOEA is also used as set mutation operator in all four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants. Note that the implementation of the set mutation step is parallelized, i.e., the μ set mutation operations can be performed in parallel as μ independent runs of the standard MOEA if the algorithm is run on a machine with more than one core. Unless otherwise stated, we always use the same parameters for all algorithms. The hypervolume indicator is computed exactly for

all bi-objective problems; otherwise, $10,000$ samples are used to approximate it; the reference point is chosen as 40^k such that all solutions of the considered problems have a positive hypervolume contribution. For comparing the algorithms, the standard MOEA runs for 500 generations with a population size of 200—the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants use the same number of function evaluations within $g_{\max} = 25$ generations where the $\mu = 10$ sets of $N = 20$ solutions each are mutated for $G = 20$ generations of the standard MOEA.

4.2 Comparison Between Four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer Variants and a Standard MOEA

To compare the four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants of Sec. 3 and the standard MOEA with the parameters described above, 30 runs are performed for each of the test problems DTLZ2, DTLZ5, DTLZ7 [9], WFG3, WFG6, and WFG9 [11] with 2, 3, and 4 objectives. Figure 3 shows the boxplots of the normalized hypervolume in the last generation, i.e., the hypervolume indicator of the set containing all single solutions in the last population. In addition, Fig. 5 shows the running times of the different algorithms on a 64bit AMD linux machine with 4 cores (2.6GHz) averaged over all 6 test problems.

There are two main observations: On the one hand, the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants are faster than the standard MOEA. On the other hand, the quality of the solution sets obtained by the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants are, in part, better than the standard MOEA in terms of hypervolume indicator values.

As to the running time, a speed-up is not surprising due to the parallel implementation of the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants. However, the speed-ups are higher than the number of cores except for the A2 variant which indicates that there will be a speed-up even on a single processor. The reason is mainly the faster hypervolume computation. To substantiate the statement that the speed-up is not only caused by the parallelization, we compare the hypervolume indicator improvements of the A1 variant and the standard MOEA over the performed function evaluations. Figure 4 shows the overall hypervolume of all solutions and the hypervolume of a randomly selected set of variant A1 together with the hypervolume of the standard MOEA averaged over 30 runs. After a certain number of function evaluations, the A1 variant outperforms the standard MOEA even for the same number of function evaluations which indicates that also a non-parallelized $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variant A1 outperforms the standard MOEA. This result is even more surprising since the standard MOEA operates on a set of 200 solutions whereas the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer operates on much smaller sets of 20 solutions only. In the latter case, the diversity between the sets of solutions is not guaranteed; in the former case, the hypervolume indicator ensures a good spread of all 200 solutions which explains the higher hypervolume in the beginning of the optimization.

As to the solution quality, we can make two observations, that are both supported by statistical tests[2]. The B1 and B2 variants obtain, statistically signifi-

[2] We used the non-parametric Kruskal-Wallis test followed by the Conover-Inman procedure with a p-value of 0.01 as described in [6] on p. 288ff.

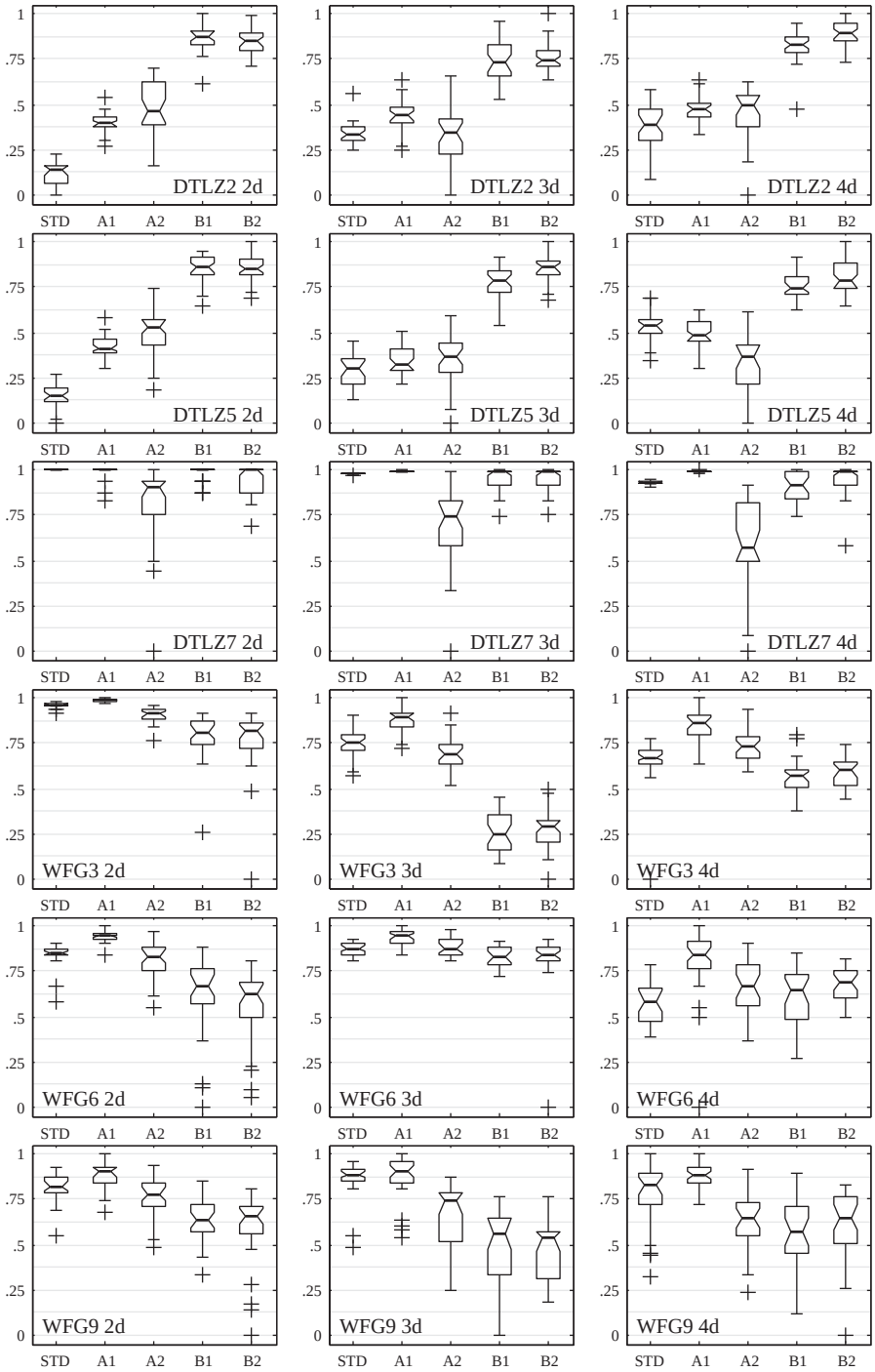

Fig. 3. Box plots of the normalized hypervolume indicator values for the four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants and the standard MOEA (STD) for six test problems with 2 (left), 3 (middle), and 4 (right) objectives. Higher values are better.

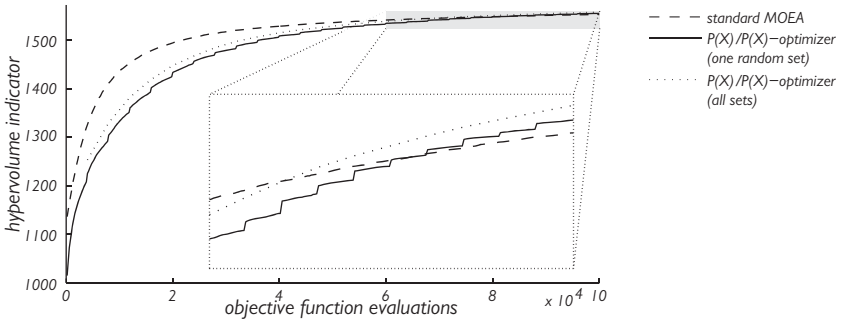

Fig. 4. Hypervolume indicator values of the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variant A1 in comparison to the standard MOEA averaged over 30 runs over time. For the A1 variant, both the overall hypervolume of all solutions and the indicator value of one randomly picked set are shown. The insert shows a detailed view of the last time period.

cantly, better hypervolume values than the standard MOEA on all DTLZ2 and DTLZ5 instances. No general conclusion over all problems can be made for the A2, B1, and B2 variants. The A1 variant, however, yields for 16 of the 18 problems better results than the standard MOEA (except for 4-objective DTLZ5 and 2-objective DTLZ7). Hence, the A1 variant is used in all further investigations.

The huge differences between the DTLZ and the WFG problems for the different $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants may be caused by the different characteristics of elitism: a good solution is more likely to be contained in *all* solution sets after recombination within the variants A2, B1, and B2 in comparison to the A1 variant, i.e., the diversity is lower. In addition, the diversity of solutions is also higher in the A1 variant because of its random mating selection. This low diversity between single solutions might be the reason why the three variants A2, B1, and B2 are not performing as good as the A1 variant on the WFG problems. For the DTLZ problems, however, the small diversity seems to cause no problems for the search due to the structure of the problems.

4.3 Comparing Different Mutations on Sets

In order to study the influence of the parameter G, i.e., the length of a mutation step, we run the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variant A1 with different values for G—all other parameters are kept the same except the number of generations which is changed to keep the overall number of objective function evaluations the same. Figure 6 shows the normalized hypervolume values averaged over 30 runs together with the standard deviation. Although the influence of G is small compared to the hypervolume of the standard MOEA, we observe a tendency towards better results if the mutation length is smaller. This gives evidence that the used set recombination is a powerful operator. However, using the set recombination more frequently results in a higher running time. Although further investigations on the choice of G are needed, our choice makes a first compromise between running time and solution quality.

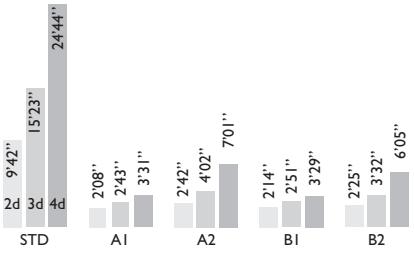

Fig. 5. Averaged running times of the four $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variants and the standard MOEA.

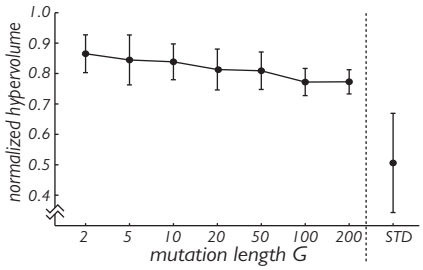

Fig. 6. Comparison between different set mutations with respect to the number G of generations (mutation length).

4.4 Application to Wireless Sensor Networks Deployment

Woehrle et al. [19] tackled the problem of placing wireless sensor nodes to monitor certain regions of interest by using a MOEA based on a new mutation operator. This problem of finding a sensor network deployment with a minimal number of nodes that minimizes the transmission error probability while the constraint of covering a certain region of interest is fulfilled, here, serves as an example application. Compared to the test problems investigated above, the evaluation time of a single solution is long: one evaluation takes up to several seconds per solution in comparison to milliseconds for the test problems.

When comparing the $\mathcal{P}(X)/\mathcal{P}(X)$-optimizer variant A1[3] and the standard MOEA with the same number of $2,000$ function evaluations exemplary for one run[4], it turns out that A1 outperforms the standard MOEA in terms of hypervolume. The indicator value of the union of all solutions sets in the last population equals 28.376 for the A1 variant compared to 26.595 for the standard MOEA (reference point at $(50, 1.1)$). Also the plot of all achieved non-dominated solutions in objective space (Fig. 7) indicates that the solutions found are of higher quality for a decision maker. With respect to the running time, the A1 variant reaches a speed-up of about 3 on a 2-processor machine with 4 cores.

5 Conclusions

This paper has demonstrated that maintaining a population of solution sets in combination with appropriate set variation and selection operators can have advantages over classical MOEAs—in a setting where the hypervolume is the set measure to be optimized. The experimental results not only show that the

[3] As parameter values, $G = 2$, $\mu = 4$, $N = 10$, and $g_{\max} = 50$ are used. Furthermore, the mutation and recombination operators on single solutions within the set mutation as well as the objective functions are implemented as described in [19].

[4] We are aware of the small significance of one run, but know that changing random seeds or problem instances does not change the results qualitatively—presenting results for more runs would, however, lengthen the paper beyond the page limit.

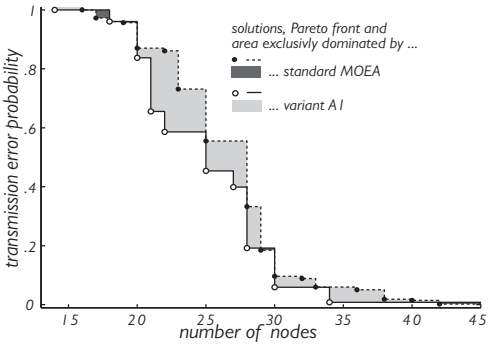

Fig. 7. Illustration of non-dominated points of the wireless sensor network application after 2,000 function evaluations.

quality of the generated Pareto set approximations can be largely improved, but that also the overall computation time can be reduced. As to the former, set recombination seems to play a major role, while the latter is mainly because the hypervolume indicator is faster to compute for small solution sets.

The present study represents just a first step towards evolutionary algorithms for sets and there are different promising directions for future research. In particular, the choice of the parameters (solution set size, population size, etc.) and the effects of different set operators need to be investigated. Moreover, it would be worthwhile to see whether similar results can be observed for other types of set optimization criteria.

Acknowledgments

Dimo Brockhoff has been supported by the Swiss National Science Foundation (SNF) under grant 112079. Johannes Bader has been supported by the Indo-Swiss Joint Research Program IT14.

References

1. F. J. Aherne, N. A. Thacker, and P. I. Rockett. Optimising Object Recognition Parameters using a Parallel Multiobjective Genetic Algorithm. In *Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, pages 1–6. IEEE Press, 1997.
2. J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, November 2008.
3. N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal on Operational Research*, 181:1653–1669, 2007.
4. J. Branke, H. Schmeck, K. Deb, and M. Reddy. Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation. In *Congress on Evolutionary Computation (CEC 2004)*, pages 1952–1957. IEEE Press, 2004.

5. C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.
6. W. J. Conover. *Practical Nonparametric Statistics*. John Wiley, 3 edition, 1999.
7. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.
8. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *LNCS*, pages 849–858. Springer, 2000.
9. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. In *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pages 105–145. Springer, 2005.
10. T. Hiroyasu, M. Miki, and S. Watanabe. The new model of parallel genetic algorithm in multi-objective optimization problems—divided range multi-objective genetic algorithm. In *Congress on Evolutionary Computation (CEC 2000)*, pages 333–340. IEEE Press, 2000.
11. S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.
12. C. Igel, N. Hansen, and S. Roth. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 2007.
13. J. Lee and P. Hajela. Parallel Genetic Algorithm Implementation in Multidisciplinary Rotor Blade Design. *Journal of Aircraft*, 33(5):962–969, 1996.
14. M. Mezmaz and N. Melab and E.-G. Talbi. Using the Multi-Start and Island Models for Parallel Multi-Objective Optimization on the Computational Grid. In *eScience*, page 112. IEEE Press, 2006.
15. C. Poloni. Hybrid GA for Multi-Objective Aerodynamic Shape Optimization. In *Genetic Algorithms in Engineering and Computer Science*, pages 397–416. John Wiley & Sons, 1995.
16. H. Sawai and S. Adachi. Effects of Hierarchical Migration in a Parallel Distributed Parameter-free GA. In *Congress on Evolutionary Computation (CEC 2000)*, pages 1117–1124, Piscataway, NJ, 2000. IEEE Press.
17. T. J. Stanley and T. Mudge. A Parallel Genetic Algorithm for Multiobjective Microprocessor Design. In *International Conference on Genetic Algorithms*, pages 597–604. Morgan Kaufmann Publishers, 1995.
18. E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. Coello Coello. Parallel Approaches for Multiobjective Optimization. In J. Branke et al., editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 349–372. Springer, 2008.
19. M. Woehrle, D. Brockhoff, T. Hohm, and S. Bleuler. Investigating Coverage and Connectivity Trade-offs in Wireless Sensor Networks: The Benefits of MOEAs. TIK Report 294, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, October 2008. accepted for publication at MCDM 2008 conference.
20. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.
21. E. Zitzler, L. Thiele, and J. Bader. SPAM: Set Preference Algorithm for Multiobjective Optimization. In *Conference on Parallel Problem Solving From Nature (PPSN X)*, pages 847–858. Springer, 2008.